ITSE301 Logic Programming

Spring 23-24

5-5-2024

العمليات الحسابية

Arithmetic العمليات الحسابية

- ❖ Introduce Prolog`s built-in abilities for performing **arithmetic**

Arithmetic in Prolog

- Prolog provides a number of basic arithmetic tools

Arithmetic	Prolog
2 + 3 = 5	?- 5 is 2+3.
3 x 4 = 12	?- 12 is 3*4.
5 – 3 = 2	?- 2 is 5-3.
3 – 5 = -2	?- -2 is 3-5.
4 ÷ 2 = 2	?- 2 is 4/2. or 4 div 2.
1 is the remainder when 7 is divided by 2	?- 1 is 7 mod 2.

Example queries

```
?- 10 is 5+5.
true

?- 4 is 2+3.
false

?- X is 3 * 4.
X=12



?- R is 7 mod 2.
R=1
```

Defining predicates with arithmetic

```
addThree(X, Y):-
    Y is (X+3).
```

```
?- addThree(1, R).
R=4


?- addThree(2,R).
R=5
```

Defining predicates with arithmetic

```
addThreeAndDouble(X, Y):-
    Y is (X+3) ∗ 2.
```

```
?- addThreeAndDouble(1,R).
R=8


?- addThreeAndDouble(2,R).
R=10
```

A closer look

- Note that expressions such as 3+2, 4-7, 5/5 are ordinary Prolog terms
  - Functor: +, -, /, *
  - Arity: 2
  - Arguments: integers
  - 3+2 ➔ +(3,2)

A closer look

```
?- X = 3 + 2.
```

A closer look

```
?- X = 3 + 2.
X = 3+2



?-
```

A closer look

```
?- X = 3 + 2.
X = 3+2
yes

?- 3 + 2 = X.
```

A closer look

```
?- X = 3 + 2.
X = 3+2
yes

?- 3 + 2 = X.
X = 3+2
yes

?-
```

The is/2 predicate

- To force Prolog to actually evaluate arithmetic expressions, we have to use

  is

  just as we did in the other examples
- This is an instruction for Prolog to carry out calculations

The is/2 predicate

```
?- X is 3 + 2.
```

The is/2 predicate

```
?- X is 3 + 2.
X = 5
yes

?-
```

The is/2 predicate

```
?- X is 3 + 2.
X = 5
yes

?- 3 + 2 is X.
```

The is/2 predicate

```
?- X is 3 + 2.
X = 5
yes

?- 3 + 2 is X.
ERROR: is/2: Arguments are not sufficiently instantiated

?-
```

The is/2 predicate

```
?- X is 3 + 2.
X = 5
yes

?- 3 + 2 is X.
ERROR: is/2: Arguments are not sufficiently instantiated

?- Result is 2+2+2+2+2.
```

The is/2 predicate

```
?- X is 3 + 2.
X = 5
yes

?- 3 + 2 is X.
ERROR: is/2: Arguments are not sufficiently instantiated

?- Result is 2+2+2+2+2.
Result = 10
yes
+(2,+(2,+(2,+(2,+(2,2))))).
+(2,8).
```

Restrictions on use of is/2

- We are free to use variables on the right hand side of the **is** predicate
- But when Prolog actually carries out the evaluation, the variables must be instantiated with a variable-free Prolog term
- This Prolog term must be an arithmetic expression

Notation

- Two final remarks on arithmetic expressions
  - 3+2, 4/2, 4-5 are just ordinary Prolog terms in a user-friendly notation:
    3+2 is really **+(3,2)** and so on.
  - Also the **is** predicate is a two-place Prolog predicate

Notation

- Two final remarks on arithmetic expressions
  - 3+2, 4/2, 4-5 are just ordinary Prolog terms in a user-friendly notation:
    3+2 is really **+(3,2)** and so on.
  - Also the **is** predicate is a two-place Prolog predicate

```
?- is(X,+(3,2)).
X = 5
yes
```

Comparing Integers

- Some Prolog arithmetic predicates actually do carry out arithmetic by themselves
- These are the operators that compare integers

Comparing Integers

Arithmetic	Prolog
$x < y$	X < Y
$x \leq y$	X =< Y
$x = y$	X == Y
$x \neq y$	X \= Y
$x \geq y$	X >= Y
$x > y$	X > Y

Comparison Operators

- Have the obvious meaning
- Force both left and right hand argument to be evaluated

```
?- 2 < 4+1.
yes

?- 4+3 > 5+5.
no
```

Comparison Operators

- Have the obvious meaning
- Force both left and right hand argument to be evaluated

```
?- 4 = 4.
yes

?- 2+2 = 4.
no

?- 2+2 =:= 4.
yes
```

Comparing numbers

- We are going to define a predicate that takes two arguments, and is true when:
  - The first argument is a list of integers
  - The second argument is the highest integer in the list
- Basic idea
  - We will use an accumulator
  - The accumulator keeps track of the highest value encountered so far
  - If we find a higher value, the accumulator will be updated